TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2646 Informační technologie

Studijní obor: 1802R007 Informační technologie

Tvorba vyspělého CMS systému

Development of advanced CMS system

Bakalářská práce

Autor: Adam Hencze

Vedoucí práce: Ing. Jiří Jeníček, Ph. D.

V Liberci 18. 5. 2012

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce.

Datum

Podpis

Poděkování

Chci poděkovat mému vedoucímu práce, Ing. Jiřímu Jeníčkovi, Ph.D., za jeho praktické zkušenosti, odborné poznatky, příkladné vedení mé práce a za jeho rady z přímého používání systému, který jsem v rámci práce vyvíjel.

Abstrakt

Tvorba vyspělého CMS systému

Bakalářská práce seznamuje čtenáře s problematikou CMS systémů, nachází a analyzuje již vyřešené příklady s kompletními, veřejně dostupnými řešeními, zaměřuje se na trh a popisuje typickou distribuci a použití těchto aplikací. Práce dále porovnává funkce a vlastnosti těchto systémů a volí průnik jejich důležitých aspektů – na základě těchto existujících vědomostí z řešení a produktů volí vlastní nejvhodnější řešení pro návrh vlastní aplikace, jak podle požadavků klienta, tak podle serveru. Tento návrh se poté snaží realizovat jako webovou aplikaci, která bude postavena pomocí technologie PHP, MySQL, HTML, CSS a pomocí Javascriptu s využitím doplňkových knihoven. Součástí implementace je návrh grafického prostředí a vzhledu aplikace, návrh zobrazovací šablony a dynamické zobrazení obsahu v administrační části aplikace. Práce po implementaci vyhodnocuje své vlastnosti a společně s hodnocením bezpečnosti je porovnává s produkty, ze kterých v rešeršní části čerpá. Realizace CMS systému je úspěšně provedena a aplikace je následně testována klienty v reálném provozu.

Klíčová slova: CMS systém, blogový systém, administrační systém, webová aplikace

Abstract

Development of advanced CMS system

This bachelor paper deals with the issue of CMS systems, it finds and analyzes already solved cases and solutions, which are publicly available, it focuses on the market and describes a typical distribution and usage of these applications. Paper then compares functions and features of these systems and it chooses a unification of their important aspects – it chooses the best solution for the design of it's own application on the basis of existing knowledge from the solutions and the products, and based on the requirements of both the client, and the server. It's trying to implement this design as a web application, which will be built with PHP, MySQL, HTML, CSS and Javascript technology with usage of additional libraries. As a part of the implementation there's a design of an interactive graphical interface, design of the theme layout, design of dynamical content rendering and design of the application layout itself. After the implementation paper evaluates its features, properties and along with the evaluation of its security it compares them with features and properties of the products, which it describes in the research part. The implementation of the CMS system is successfully made and the application is tested by the clients in real traffic.

Keywords: CMS system, blog system, administration systém, web application

Obsah

Prohlášení ... 3

Poděkování ... 4

Abstrakt ... 5

Tvorba vyspělého CMS systému ... 5

Abstract ... 6

Development of advanced CMS system ... 6

Obsah ... 7

Seznam obrázků ... 9

Seznam tabulek ... 10

Seznam zkratek ... 11

Úvod ... 12

1 Rešerše ... 13

1.1 CMS systém ... 13

1.2 Pohled na trh ... 14

1.3 Porovnání vlastností existujících systémů ... 15

Wordpress ... 15

Joomla ... 17

Blogger ... 17

Dodatek – GNU GPL ... 17

2 Návrh ... 18

2.1 Shrnutí požadavků ... 18

2.2 Systém ... 19

Administrace ... 19

Šablona ... 19

2.3 Grafické prostředí ... 20

Prostředí administrace ... 20

Šablona 20
Přihlášení do systému 20
Palubní deska 21
Logo 21
3 Realizace 22
3.1 Využité technologie 22
Strana serveru 22
Strana klienta 23
3.2 Implementace 25
Databázový model 25
Šablona 26
Přihlášení do systému 27
Administrátor a redaktor 28
Systémový sledovač 28
Systémové přehledy 29
Rychlý obsah 30
Zobrazení obsahu 30
Práce s multimédii 31
Vyhledávání 31
Koš 31
Dodatek – šifrovací metody MD5 a SHA1 32
3.3 Uživatelská dokumentace 32
4 Vyhodnocení 33
4.1 Bezpečnost 33
4.2 Srovnání možností s ostatními systémy 34
Závěr 35
Seznam literatury 36

Seznam obrázků

Obrázek 1: Grafické rozhraní administrační části systému Wordpress 16
Obrázek 2: Grafické logo aplikace..... 21
Obrázek 3: Databázový model aplikace..... 25
Obrázek 4: Výchozí vzhledová šablona..... 26
Obrázek 5: Přihlašovací formulář systému 27
Obrázek 6: Hlavní stránka administrace 29
Obrázek 7: Zobrazení článků v administraci 30

Seznam tabulek

Tab. 1: Přehled pravomocí uživatelských rolí systému 28

Seznam zkratek

[1] CMS – Content Management System, *Systém pro správu obsahu*

[2] PHP – Hypertext Preprocessor, *Hypertextový preprocesor*

[3] MySQL – My Structured Query Language, *Můj strukturovaný databázový jazyk*

[4] HTML – HyperText Markup Language, *Značkovací jazyk pro hypertext*

[5] URL – Uniform Resource Locator, *Jednotný lokátor zdrojů*

[6] GPL – General Public License, *Obecná veřejná licence*

[7] CSS – Cascading Style Sheets, *Kaskádové styly*

[8] WYSIWYG – What You See Is What You Get, „*Dostaneš to, co vidíš*“

[9] AJAX – Asynchronous Javascript and XML, *Asynchronní Javascript a XML*

[10] XML – Extensible Markup Language, *Rozšiřitelný značkovací jazyk*

[11] MD5 – Message Digest 5, *Výběr zprávy 5*

[12] SHA1 – Secure Hash Algorithm 1, *Zabezpečovací hashovací algoritmus 1*

[13] IP – Internet Protocol, *Internetový protokol*

[14] SSL – Secure Sockets Layer, *Vrstva bezpečných socketů*

[15] HTTPS – *Zabezpečený protokol přenosu hypertextu*

Úvod

Tato bakalářská práce se zabývá vývojem CMS systému primárně určeného pro blogové webové stránky a blogové portály. CMS systém je vlastně sofistikovaně propracovaná dynamická webová stránka, aplikace, která se stará o obsah, se kterým může obecně manipulovat (upravovat ho, mazat, vkládat). Internet je v dnešní době tvořen téměř výhradně dynamickým obsahem, který je potřeba spravovat, a právě o to se tyto CMS systémy starají. Přitom se může se jednat nejen o aplikaci pro správu blogového obsahu, ale třeba i o velké novinářské portály, diskusní fóra, fotogalerie, sociální portály apod. Aplikace má definované základní nutné funkce, které pro své fungování potřebuje.

Chci navrhnout a implementovat co nejefektivnější a zároveň jednoduché řešení pomocí typických funkcí CMS systémů a pomocí vlastností již hotových řešení tak, aby byl systém co nejblíže jednoduchosti a intuitivnosti pro typické nenáročné uživatele. K tomu potřebuji provést průzkum trhu s již hotovými a úspěšnými aplikacemi, které musím analyzovat a na jejich základě vytvořit vlastní návrh, který bude tvořen průnikem jejich nejzajímavějších funkcí a mých vlastních nápadů.

1 Rešerše

1.1 CMS systém

CMS[1] systém je software, který je dnes v drtivé většině případů realizován jako webová aplikace, která běží na webovém serveru. Pro jednoduchost jde vlastně o propracovanou webovou stránku napsanou pomocí některého ze skriptovacích jazyků, která má za úkol poskytovat správu všech druhů webového obsahu, tedy článků, dokumentů, uživatelů, kategorií apod.

Hlavním rozdílem mezi CMS systémem a běžnou statickou webovou stránkou je v samotném způsobu manipulace s obsahem, který je v případě dynamického přístupu CMS uložen v databázi a lze s ním proto skrz grafické webové prostředí manipulovat efektivněji. V případě statických stránek musí uživatel neustále přepisovat obsah stránek přímo, a to otevřením souboru se stránkou manuálně. Tento způsob se pro většinu případů webových stránek nehodí, ale například pro jednoduchý prezentační web, kde se data nebudou tak často měnit, je CMS systém zbytečný.

CMS nejčastěji funguje jako část dynamické aplikace, která již obsahuje několik typů šablon pro vzhled stránek jako jednu část (tu vidí běžný návštěvník stránek), a administrační rozhraní samotné jako část druhou (do té má přístup pouze správce webových stránek/systému, popřípadě ještě další uživatelé, například redaktoři webového magazínu). Její základní funkce jsou obsaženy v jádru, které si uživatel určitým způsobem nainstaluje (tyto způsoby jsou v zásadě stejné a jejich popisem se zabývám dále).

Takto nainstalovaný systém neumí pouze spravovat obsah, jeho největší výhodou je velké množství už napsaných a vytvořených doplňků, které si uživatel může přímo od vývojářů (nezávislých vývojářů nebo přímo od vývojářů aplikace) stáhnout a do systému doinstalovat. Aplikace je tak velmi adaptibilní potřebám uživatele a lze ji využít téměř na cokoliv. Funkční rozsah těchto doplňků je totiž velmi široký, CMS systém lze potom využít například pro správu několika úrovňového diskusního fóra, dynamické fotogalerie, elektronického obchodu apod.

[1] Content Management Systém, v češtině „systém pro správu obsahu“, označuje se i jako „redakční systém, nebo „publikační systém“

1.2 Pohled na trh

V dnešní době se na trhu vyskytuje mnoho volně stažitelných nebo placených profesionálních CMS systémů. Jejich návrhy jsou různé, nicméně jádro zůstává v zásadě stejné. Ve všech případech se jedná o správu obsahu a všech věcí, které se k němu váží, správa uživatelů, kategorií a typů obsahu, nahrávání multimediálního obsahu a mnoho dalších, pro většinu systémů standardních věcí.

Většina z těchto hotových řešení funguje na podobném principu. Uživatel si po výběru systému, který chce použít, z jeho oficiálních stránek stáhne zabalený balíček, ve kterém se nachází celý systém ve výchozí konfiguraci. Každý systém má přirozeně jiné požadavky na server (tedy stroj, na kterém poběží), například určitou verzi programovacího jazyka, verzi a typ databáze, určitou velikost dostupné paměti apod. Pokud uživatelský server tyto požadavky splňuje, uživatel nahraje rozbalený systém na server.

Aplikace má ve výchozím nastavení definovaného průvodce, který uživatele skrz grafické prostředí navede postupně celou konfigurací systému. Uživatel zde musí vyplnit například přístupové údaje do databáze (to je nejdůležitější položka, bez které se aplikace nerozběhne), název a popis aplikace, svůj e-mail pro bezpečnostní účely, a samozřejmě si zde i určí svoje uživatelské přístupové jméno a heslo do samotného systému.

Pokud je systém volně dostupný a stažitelný, pak je jeho instalace logickým řešením. Já cílím spíše na skupinu klientů, kteří si zadají u vývojáře nějaký typ práce, a příslušné nastavení s konfigurací systému je na programátorovi. Proto je jakákoliv grafická instalace zbytečná, pro správnou konfiguraci systému bude stačit otevřít konfigurační soubor, kde lze přepsat potřebné údaje.

Po této konfiguraci už aplikace běží na serveru v ostrém provozu s výchozím obsahem, což může být například jeden úvodní článek s nějakým generickým textem (například „Lorem ipsum“, standartní výplňový text), „Hello world!“ nebo něco podobného.

1.3 Porovnání vlastností existujících systémů

Zkoumal jsem nejrozšířenější aplikace, které používá největší počet uživatelů a které mají kolem sebe největší komunitu. Všechny jsem se pokusil co nejlépe osvojit, dobře vyzkoušet a vzít si z nich pouze ty věci, které potom použiji do návrhu mého vlastního CMS systému. Bylo třeba najít jakýsi kompromis mezi jednoduchostí a funkčností. Můj návrh měl být intuitivní, zároveň měl ale zachovat všechny funkce a měl být především bezpečný.

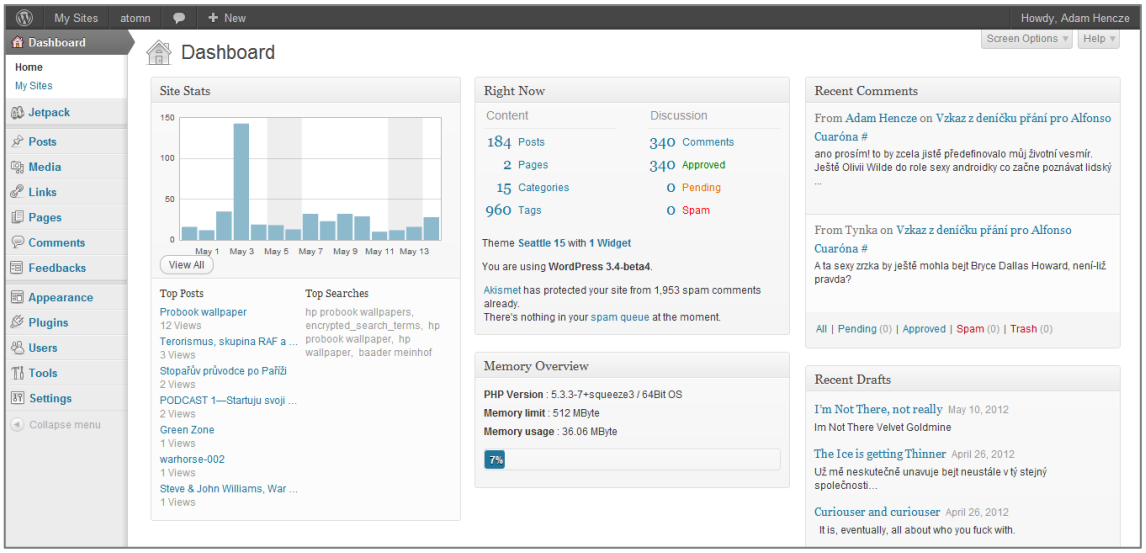
Wordpress

Wordpress je nejkomplexnější bezplatné řešení na trhu s obrovským množstvím doplňků, které píší různí vývojáři a komunita kolem systému obecně. Systém je možné využít v mnoha způsobech, v základu se používá jako osobní blog, díky jednoduchému vytváření šablon a doplňků lze ale systém použít téměř na cokoliv, od prezentačních webových stránek po komplexní redakční systém novinářského portálu, nebo elektronického obchodu. Nová verze přichází jednou za půl roku, menší aktualizace společně s bezpečnostními záplatami jsou vydávány zhruba měsíčně, pokud je potřeba tak i častěji (objeví-li například někdo bezpečnostní chybu). Výhodou systému je také zmiňovaná komunita, takže pokud si uživatel neví rady, téměř jistě na internetu vše dohledá, protože někdo před ním už podobnou chybu také řešil. Od verze 3 umí Wordpress spravovat velké sítě jednotlivých „podsystémů“, takže v rámci jedné instalace můžete spravovat například dva blogy, redakční portál a elektronický obchod, což je neskutečně výhodné a systém to povýšilo o několik úrovní výše. Aplikace je distribuována zdarma přímo z oficiálních stránek vydavatele, odkud se dá zabalená stáhnout do počítače a pomocí jednoduché instalace rozběhnout na serveru, kde musí běžet PHP[2] verze 5.2.4 nebo vyšší a MySQL[3] verze 5.0 nebo vyšší. Toto kritérium splňuje drtivá většina strojů. Bohužel je ale Wordpress jedním z nejsložitěji napsaných systémů, aplikace je rozdělena v souborech, kterých je v balíku tisíce, její jádro, psané v PHP, se generuje velmi dlouho a načítání komplexních blogů proto trvá v porovnání s ostatními systémy velmi dlouho (tento nedostatek lze vyřešit instalací doplňku, který z dynamické aplikace vytvoří na serveru samostatné generované HTML[4] stránky,

[2] Hypertext Preprocessor, v češtině „Hypertextový preprocesor“
[3] My Structured Query Language, v češtině „Můj strukturovaný databázový jazyk“
[4] HyperText Markup Language, v češtině „Hypertextový značkovací jazyk“

jejichž načítání je velmi rychlé). Pro „hezká“ URL[5] musí mít ještě server zapnutý mod_rewrite.

Inspiroval jsem se správou administrace jako takové a grafickým rozhraním základní šablony, kterou Wordpress definoval moderní blogovou éru. Wordpress je nejrozšířenějším CMS systémem s více než 15 % používání na internetových stránkách celého světa [8]. Líbí se mi její jednoduchost, funkčnost a neustálá aktivita kolem systému, který je tak aktuální a na poli CMS systémů profesionální.



Obrázek 1: Grafické rozhraní administrační části redakčního systému Wordpress

[5] Uniform Resource Locator, česky „Jednotný lokátor zdrojů“

Joomla

Obecnější systém s širší možností využití. Má velice kvalitní doplňky například pro internetový obchod, komplexní diskuzní fórum nebo kalendář. Spoustu uživatelů ale odradí svou neintuitivností a složitým ovládáním, které je sice dobře škálovatelné, ale člověk se v něm musí umět dobře vyznat. Joomla používá úplně jinou strukturu administrace a složení webových stránek [6], kde propojuje odkazy s jednotlivými menu a ty až pak s jednotlivými stránkami. Je to složité, ale spoustě řešení to vyhovuje a je to v některých případech i výhodné. Touto cestou jsem se rozhodl nejít, nelíbí se mi neintuitivní aplikace, u kterých není na první pohled jasné, jak co funguje, a to je přesně to, co Joomla představuje. Vydávání aktuální verze probíhá jednou za půl roku.

Blogger

Systém, který je vyvíjen společností Google a který do kategorie CMS úplně nezapadá, přesto jsem se inspiroval jeho správou článků a několika dalšími drobnostmi. Má uzavřený přístup do systému, působí jako služba, ne jako samostatný stažitelný balíček. Má velkou spoustu možností volby šablon a vzhledu, většině uživatelů se líbí čistý a minimalistický design, jinak je proti zbytku zkoumaných řešení primitivnější. Možná je právě to ale důvod, proč je Blogger společně s Wordpressem nejpoužívanější platformou pro blogové systémy. Stejně jako Wordpress umožňuje i Blogger vytvořit malou blogovou síť, kde lze spravovat jednotlivé blogy a různě je propojovat a přesměrovávat.

Dodatek – GNU GPL

Joomla a Wordpress jsou licencovány pod licencí GNU GPL[6], což je ve zkratce licence pro svobodný software, která vyžaduje odvozená díla dostupná pod stejnou licencí. Tento typ licencování je v internetovém i ryze počítačovém softwaru velmi rozšířený a oblíbený, jeho typickým příkladem je distribuce operačního systému Linux.

[6] GNUs Not Unix!, česky „GNU Není Unix!“, General Public License, česky „všeobecná veřejná licence“

2 Návrh

Po průzkumu trhu s webovými aplikacemi a s již existujícími řešeními jsem začal pracovat na vlastním CMS systému, kterému jsem na začátku potřeboval navrhnout základní funkce. Do návrhu jsem použil všechny požadavky na aplikaci, ty, které jsem získal rešerší a ty, které jsem vymyslel sám.

2.1 Shrnutí požadavků

Navrhuji systém pro správu obsahu, to je tedy ta nejzákladnější funkce ze všech, vkládání, úprava a zobrazení obsahu. Ten je dostupný v databázovém úložišti přes spojení realizované skriptovacím jazykem, který si zvolím. Obsah samotný budou spravovat lidé, což mohou být správci a všemožní editoři, proto jsem implementoval funkci správy uživatelů, kteří mají dva druhy práv přístupu do systému (omezující pro redaktory a kompletní pro administrátora) a které lze přidávat, mazat a upravovat jejich vlastnosti. To samé platí pro správu multimediálního obsahu (obrázky), který musí ještě splňovat kritéria k tomu, aby byl na server nahrán (jedná se hlavně o definovanou maximální velikost a typ obrázku/souboru). Obsah bude nutné kategorizovat, proto jsem navrhl jednoúrovňový kategorizační systém, který lze spravovat stejně jako všechny ostatní prvky.

Webové stránky mají také další, v dnešní době základní a velmi nutný požadavek – vyhledávání obsahu pomocí textového pole. To byl jeden z hlavních důvodů, proč jsem se rozhodl použít systém klíčových slov, která si správce k obsahu může dopsat a tím tak pomoci při vyhledávání. V dnešní době sociálních sítí je taky velice populární ke všemu vyjadřovat svůj názor, obsah komentovat, takže komentáře jsou dalším typem obsahu, který lze spravovat.

Za jednu z nejdůležitějších věcí považuji statistiky, díky kterým má správce systému jasný přehled o svém obsahu a o tom, co nového se na stránkách v jeho nepřítomnosti stalo. S touto věcí souvisí funkce, na kterou jsem u ostatních systémů většinou nenarazil a kterou jsem si sám navrhl, jde o sledovač aktivity. Ukládá se do něj jakýkoliv přístup do databáze s účelem upravit data a opět se jedná o pomocníka pro správce systému. Data sledovače nepůjde nijak upravovat, půjde o prostého informačního správce.

2.2 Systém

Po stanovení požadavků jsem začal navrhovat architekturu systému samotného. Rozvrhl jsem systém do dvou základních zobrazovacích částí, první je administrace samotného systému (odteď „administrace“), druhá je šablona výchozího vzhledu (odteď „šablona“), na které je zobrazen obsah pro návštěvníka stránek. Některé funkce jsou pro obě tyto části společné, například vyhledávání je přítomno jak na šabloně, tak v administraci.

Administrace

Systém bude poskytovat výpis, editaci a vkládání obsahu (ať už se jedná o články, kategorie, média nebo komentáře), to jsou tedy hlavní tři pilíře práce s webovým obsahem. Editace se vztahuje ke konkrétní položce, proto ji naimplementuji společně do výpisu obsahu a tyto dvě funkce spojím (bude ještě samozřejmě možné položku vyhodit do koše a potom kompletně smazat). Administrace tedy bude sestávat z nějakého hlavního menu, které bude mít tyto dvě (výpis a vkládání obsahu) hlavní funkce jako odkazy společně s hlavní palubní deskou, kde se budou zobrazovat statistiky a přehledy o systému. Pro škálovatelnost systému a změny v konfiguraci ještě přidám odkaz do stránky s hlavním systémovým nastavením. Práce v administraci musí být účinná, svižná a dynamická, proto vyřeším výpis a filtraci obsahu pomocí nějaké moderní technologie, která může pracovat na pozadí klientovi práce a posílat mu aktuální data přímo do prohlížeče bez nutnosti obsah znovu manuálně načítat. Tato funkce umožní zrychlení práce.

Šablona

Návrh šablony je proti tomu o dost jednodušší, přitom musí být ale funkční a musí upoutat, protože právě šablona je to, co uživatel při návštěvě stránek na obrazovce uvidí. Šablona pro blog musí obsahovat výpis autorových článků, to je její primární funkce. Chci na ní dostat ještě určitě funkci vyhledávání, odkazy na jednotlivé stránky a do administrace (pro správce). Taky zde bude výpis kategorií. Neměla by se pomalu načítat, proto ji navrhnu graficky jednoduchou. Je také dobré na ní zobrazit nějaké informace o autorovi a vlastní název blogové stránky, ta bude zvýrazněná a vypíšu jí do hlavičky stránky.

Aplikaci jsem se rozhodl nazvat Guardian (česky „Ochránce“), a to především kvůli slovnímu významu. Snažím se také cílit na podvědomí zákazníka/uživatele, jakmile má webová aplikace jasný název a grafické logo, vypadá mnohem seriózněji a dá se identifikovat jako značka. Po teoretickém návrhu jsem v grafických editorech začal navrhovat vzhled systému.

2.3 Grafické prostředí

Snažil jsem se o jednoduchý design s lehkými odstíny barev, které nebudou uživatele mást a umožní mu dlouhodobější práci se systémem, například když bude psát delší článek. Je to výhodné i v rychlejším načítání stránek, stroj klienta nemusí počítat všechny složité elementy kaskádových stylů.

Prostředí administrace

Vzhled systému musí být především intuitivní, uživatel by měl hned po přihlášení vědět, jak zhruba systém funguje a přes jaký odkaz se dostane tam, kam potřebuje. Začal jsem s návrhem informační lišty, která by měla obsahovat název blogu a informace o uživateli, pod kterou se bude vyskytovat menu s obsahem, který bude na aktuálním výběru z menu záviset.

Šablona

Rozhodl jsem se pro jednoduchost navrhnout šablonu s minimalistickým vzhledem, který odpovídá dnešním webovým standardům. Mou cílovou skupinou jsou hlavně lidé, co chtějí systém používat jako blog, proto jsem v grafických programech kreslil dvousloupcové vzhledy (s poměry sloupců 7 : 3, širší pro obsah, užší pro menu, archiv a další položky) s jednoduchou hlavičkou a patičkou. Barvy a vzhled elementů jsem zvolil podobný, jako mělo staré výchozí téma systému Wordpress, neboť mi přišlo přehledné a graficky úhledné.

Přihlášení do systému

Šablona a administrace jsou dvě oddělené zóny, které jsou logicky striktně oddělené. Bylo proto nutné vymyslet jakýsi most mezi těmito stránkami, který rozhodne, má-li uživatel práva na vstup do systému. To bude realizovat přihlašovací formulář, který ověří uživatelovo jméno a heslo, pokud nalezne v databázi uživatelů shodu, proběhne přihlášení a přesměrování do prostředí administrace. To bude pro bezpečnost dobré ověřovat při každé akci uživatele v administraci.

Aby správce stránek nemusel pořád dokola vkládat své údaje, využil jsem funkci zapamatování údajů, která využívá cookies, tedy údajů, které se uloží v počítači klienta. Pro bezpečnost se tyto údaje jednou za čas smažou, aby se zabránilo možnému bezpečnostnímu nedopatření.

Palubní deska

Je zobrazením systémových přehledů a hlavní stránky administrace. Sem chci implementovat všechny možné statistiky včetně funkce rychlého obsahu, což bude něco jako zápisník s možností ihned publikovat obsah, který zde autor napíše (a nemusí tak chodit na psaní nového článku, jedná-li se o malou publikaci nebo je-li autor ve spěchu). Podle počtu komentářů budu také hodnotit oblíbenost příspěvků a záznamů v databázi včetně zobrazení nejnovějších komentářů k příspěvkům.

Palubní deska je taky hlavní stránkou a tím prvním, co správce v administraci uvidí. Všechny funkce v administraci by měli být správci hned jasné, stejně tak jako by ho měla úvodní stránka graficky navnadit, uživatel by se měl do administrace rád vracet a nebrat ji jako nezbytnou součást práce.

Logo

Pro zajímavost jsem navrhl i grafické logo systému, všechny velké značky jsou v dnešní době identifikovatelné svým logem. Aplikace díky němu také vypadá více seriózně. Logo se potom zobrazí v hlavičce administrace a také na přihlašovací obrazovce. Při samotném návrhu loga jsem vycházel z již vymyšleného názvu pro systém, tedy Guardian, a vzal v potaz první písmeno. To jsem pomocí nástrojů štětce a maskování zasadil do kruhového elementu. Kruh je navíc symbolem jednoty a věčnosti, což se vždycky hodí.

Obrázek 2: Grafické logo aplikace

3 Realizace

Po dokončení návrhu jsem nejprve nainstaloval serverový software Apache, který je pro mé řešení ideální platformou. Aktualizoval jsem všechny technologie, které popisuji níže a spustil si tak vlastní testovací stroj, pro který jsem mohl začít psát kód jádra systému. Výhodou takovéhoto testovacího prostředí je libovolná konfigurace samotného serveru, to není u dnešních poskytovatelů webových serverů zvykem (zpravidla u nich platí všemožná omezení, na vlastním stroji se dá ale nastavit cokoliv dle potřeby aplikace).

3.1 Využité technologie

Technologie a jazyky dělím podle toho, na jakém místě pracují. Skriptovací jazyk společně s databází pracuje přímo na stroji serveru, který potom pouze odešle klientskému počítači výsledek, kdežto jazyk HTML nebo Javascript pracuje na stroji klienta.

Strana serveru

Dynamičnost aplikace vyžaduje volbu z několika možností dnešních moderních programovacích jazyků. Velice se mi líbila možnost Ruby (moderní skriptovací jazyk původem z Japonska), mezi jeho vlastnosti patří skvělá škálovatelnost a rychlost, bohužel pro jeho chod je třeba server, na kterém je Ruby nainstalované. To je pro většinu dnešních serverových strojů, na kterých bych mohl mít aplikaci nahranou, velký problém. Nakonec jsem po spočítání celkové náročnosti psaní a komplexnosti systému zvolil jazyk **PHP** (Hypertext Preprocessor, v češtině „Hypertextový Preprocesor“), hlavně pro jeho intuitivnost a široké možnosti využití. Tento jazyk je dnes nejběžnějším a nejrozšířenějším prostředním pro webové dynamické aplikace s podporou většiny serverů.

Obsah stránek, který se skriptuje jazyk PHP, pracuje přímo na serveru, stejně jako u většiny ostatních skriptovacích jazyků. Je to procesor serverového stroje, kdo podle kódu vyhodnotí požadavek, který mu od klienta přijde, a podle toho odešle patřičnou informaci zpět klientskému počítači.

Nejdůležitějším prvkem dynamické webové aplikace je samozřejmě uložiště dat, v mém případě databáze. Z několika druhů jsem se rozhodl využít nejrozšířenější volně šiřitelný software s názvem **MySQL** (My Structured Query Language, v češtině „Můj

Strukturovaný Databázový Jazyk“), a to hlavně pro jeho výkon, rychlost, spolehlivost a protože je multiplatformní. Zvládá všechny typy databázových operací, umí pracovat s triggery (spouštěči), indexovat a využívá pohledy. Komunikace probíhá pomocí jazyka SQL, lze jí realizovat klasickou cestou pomocí příkazové řádky, což je pro složitější databáze nevhodné. Pro přehledné grafické prostředí komunikace s databází se dnes využívá především phpMyAdmin, já se rozhodl zvolit jednodušší, které se jmenuje Adminer a je volně stažitelný od autora Jakuba Vrány. Jedná se vlastně o jednodušší verzi zmiňovaného phpMyAdmina, je ale rychlejší a ztratil několik zbytečných funkcí.

Strana klienta

Samotné webové stránky jsem napsal standardně v jazyce **HTML**, který definuje jednotlivé elementy stránek, a pomocí **CSS**[7], kaskádových stylů, které tyto elementy formátují a stylují. Tyto dva webové jazyky definují strukturu a vzhled samotných stránek. CSS je pouze doplněk kódu HTML, bez něj jsou kaskádové styly nepoužitelné.

Skvělým doplňkem k jinak statickému HTML je **Javascript**, který jsem se rozhodl využít na pomocné funkce, dynamické zobrazování různých částí stránek a vylepšení, které zlepší uživatelský prožitek. Javascript je jazyk, který pracuje na straně klienta v prohlížeči. K tomuto jazyku jsem ještě naimplementoval dvě knihovny, které jsou v Javascriptu napsané, a to **jQuery** a **mootools**. To se například využívá i u WYSIWYG[8] editoru, pomocí kterého se píše a styluje webový obsah v administračním prostředí. O doplňku jQuery se dnes nemluví jako o doplňkové knihovně, ale jako o Javascriptovém frameworku, což je vlastně jakási struktura nad jazykem, která usnadňuje vývoj a ladění aplikace. Tento framework sice z jazyka Javascript vychází, při jeho psaní je ale odlišný. Frameworky jsou v dnešní době vysoce populární, mohu jmenovat například skvělé Nette pro jazyk PHP nebo framework Rails pro jazyk Ruby. Technologii tohoto editoru ještě popíšu dále.

S jazykem Javascript velice úzce souvisí moderní dynamický přístup k obsahu, **AJAX**[9], který využívá kombinaci jazyků Javascript a PHP. Jde o funkci zasílání PHP skriptů na server bez nutnosti odesílat typický http požadavek znovunačtením celé stránky, což je mnohdy pomalé. Administrační část musí být rychlá, proto jsem

[7] Cascadian Style Sheets, v češtině „Kaskádové Styly“
[8] What You See Is What You Get, v češtině „Dostaneš to, co vidíš“
[9] Asynchronous Javascript, v češtině „Asynchronní Javascript“

zobrazování a editaci obsahu napsal kompletně AJAXem. Toto využívám například při filtraci obsahu a díky tomu působí systém opravdu dynamicky. Nevýhodou této metody jsou vyhledávače, které obsah generovaný AJAXem neindexují, což ale mému účelu v administračním prostředí systému nevadí (není třeba ho indexovat do žádných katalogů, obsah na šabloně samozřejmě ano, tam se ale AJAX nevyužívá).

Ze serverových technik jsem se ještě rozhodl využít a zapnout **mod_rewrite**, který je doplňkem softwaru serveru (většinou se jedná o Apache webový server), který umožňuje maskování URL adresy PHP stroji. Tento mód umožňuje například využití „hezkých" URL adres, kdy uživatel vidí „`http://stranka.cz/muj-prvni-clanek`", místo nic neříkajícího „`http://stranka.cz/?page=clanky&clanekid=3`". To je opět doména dnešní moderní blogové a internetové éry, která umožňuje i efektivnější indexaci, snadnější vkládání do katalogů a lepší hodnocení vyhledávačů (a samozřejmě je to hezčí i z estetického hlediska).

Jako WYSIWYG editor jsem se rozhodl využít knihovnu **CKeditor**, kterou ve svých systémech využívá například firma IBM nebo Oracle. Tento editor obsahuje všechny obvyklé formátovací funkce, jeho hlavním účelem je být jako malý a funkčně omezený Microsoft Word, který zná a umí používat většina dnešních uživatelů počítače. Rád bych podotkl také na speciální vkládací funkci „vložit z MS Word", která zkopíruje obsah přímo z kancelářského programu od Microsoftu a zachová jeho formátování. Velká část uživatelů je totiž zvyklá si svůj obsah nejprve předepsat ve Wordu, kde si ho naformátují a nastylují, vložení tohoto obsahu potom ve většině případů nefunguje tak, jak by mělo. Tato funkce to umožňuje.

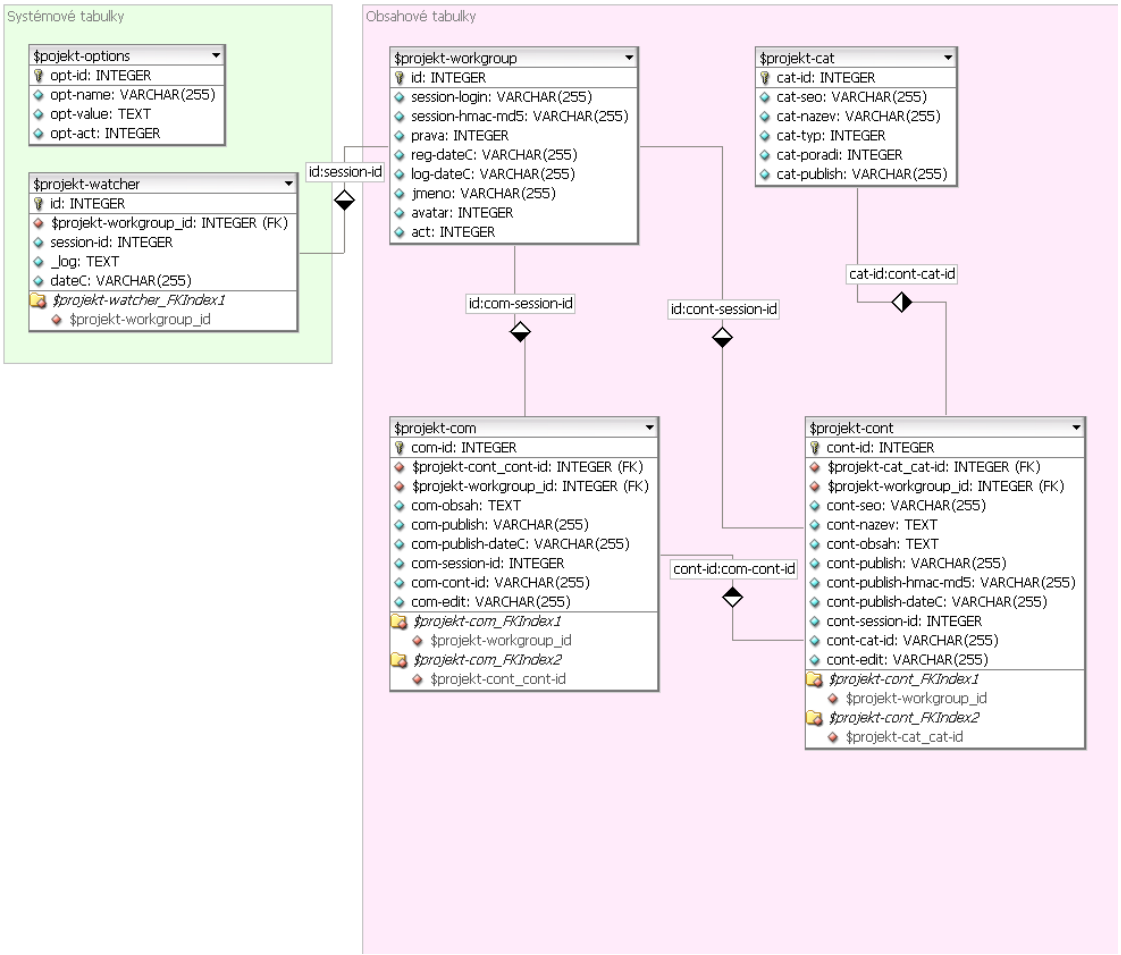
Požadavky aplikace na server jsou tedy celkem jasné – PHP alespoň verze 5.2 nebo vyšší, MySQL verze 5.0 nebo vyšší a povolené úpravy htaccess, to se využívá u zmíněného mod_rewrite. Ideálním serverovým softwarem je Apache, ale není to požadavek, pouze doporučení.

3.2 Implementace

Databázový model

Z technického hlediska šlo o první realizační krok po návrhu databázové modelu. Celý ekosystém jsem rozdělil do dvou částí, a to pro obsah a pro systém samotný. V samotném systému jsou tabulky pro nastavení a sledovač systému.

V tabulce nastavení jsou hodnoty jako adresy, složkový systém na serveru a cesty k různým zdrojovým složkám, použitá šablona a mnoho dalších nastavení. Sledovač má v sobě záznamy o akcích v systému. Obsahová část skrývá všechny čtyři obsahové tabulky, a to – uživatelskou, kategorickou, komentářovou a obsahovou. Rozhodl jsem se takto tabulky pro přehled a pro lepší správu rozdělit, mít všechno v jedné tabulce je možné, ale brzo by se začala velmi zvětšovat a mohl by mírně klesnout výkon databáze.

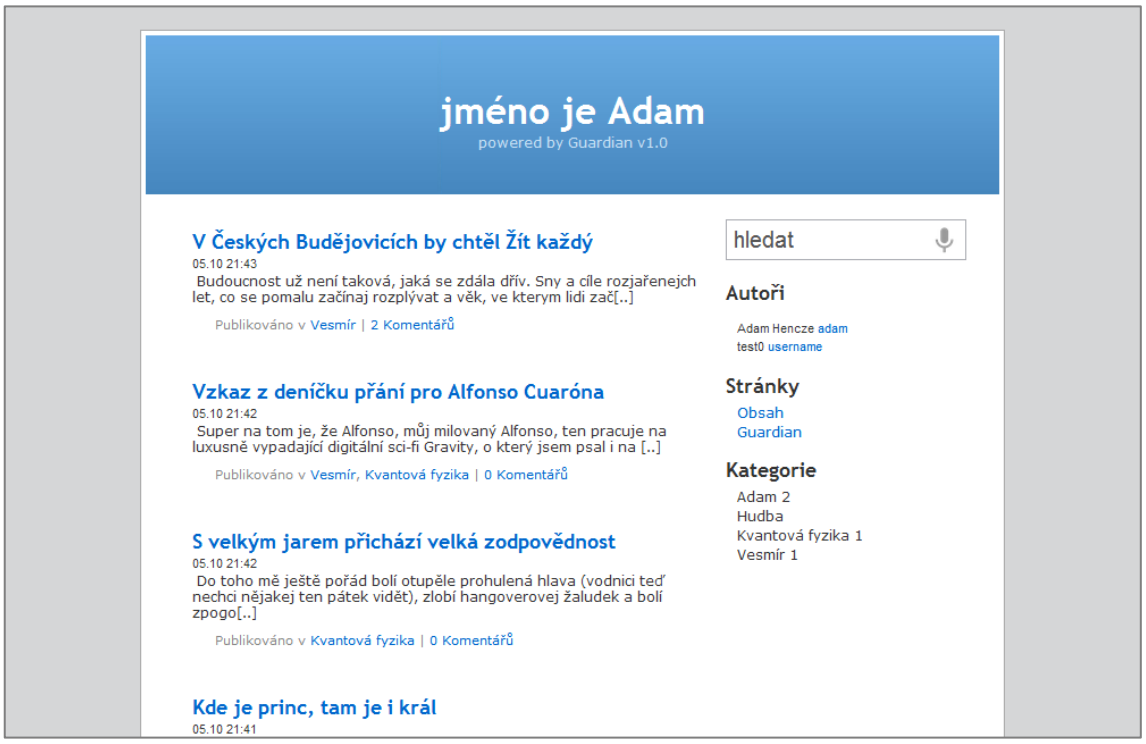


Obrázek 3: Databázový model aplikace

Šablona

Napsal jsem kód pro hlavní zobrazení, ve kterém se nachází hlavička s modrým pozadím a názvem blogu, pod kterým je vypsán název a verze systému. To je vnořeno do zobrazovacího elementu, který má bíle pozadí a černý text (zvolil jsem klasický přístup, který se dobře čte). Jako hlavní tělo jsem definoval dva sloupce, levý sloupec zobrazuje hlavní obsah, je proto širší a na obrázku níže je vidět ve výchozím zobrazení na domovské stránce (výčet článků blogu). Pravý sloupec pod vyhledávacím polem zobrazuje všechny autory blogu, všechny stránky a jednotlivé kategorie, které mají vedle sebe počet článků, které pod ně spadají.

Po klepnutí na název článku v levém sloupci se zobrazení změní a uživatel vidí obsah článku, pod kterým jsou vypsány všechny komentáře společně s komentářovým formulářem, přes který může uživatel článek komentovat (nemusí se nikam přihlašovat). O to, jaká stránka se kdy zobrazí, se stará PHP jádro, které čte URL adresu, podle které se rozhoduje.



Obrázek 4: Výchozí vzhledová šablona

Přihlášení do systému

Do administrace systému se uživatel dostane buďto přes odkaz ze šablony (je umístěn v pravém sloupci), nebo zadáním URL adresy webových stránek s přídavkem „guardian“ za zpětným lomítkem, tedy například „`http://stranka.cz/guardian`“. Pokud se uživatel ještě nepřihlašoval a neukládal si pomocí funkce „Zapamatovat na počítači“ cookies, zobrazí se mu přihlašovací stránka systému.

Nad formulářem se zobrazuje logo, které jsem pro systém navrhl, a pod ním se nachází samotný formulář, který sestává ze dvou textových vstupních prvků, kam uživatel zadá své jméno a heslo. Pomocí zaškrtávacího pole jsem realizoval funkci zapamatování údajů, která využívá již zmíněné cookies, tedy informace klienta uložené v paměti webového prohlížeče. Po klepnutí na tlačítko „Přihlásit se“ se na serveru porovnají zadané údaje s databází, pokud se najde shoda jména, porovná se vstupní heslo, které se šifruje pomocí jednostranných šifrovacích metod MD5[10] a SHA1[11], jejich bližší popis se nachází dále, prozatím stačí vědět, že jsou to jednostranné funkce, nelze je tedy přečíst zpět. Pokud se nalezne shoda i v něm, uživatel je úspěšně přihlášen. Toto porovnání se provede i po přihlášení s každou akcí uživatele, a pokud se nějakým způsobem během jeho působení v systému změní hodnota proměnné session (kde jsou jeho informace uloženy), systém ho odhlásí.

Obrázek 5: Přihlašovací formulář do systému

[10] Message Digest 5, česky „Výběr Zprávy 5“
[11] Secure Hash Algorithm, česky „zabezpečovací hashovací algoritmus“

Při nesprávných údajích formulář vypíše chybové hlášky, může dojít ke špatně napsanému heslu nebo neexistujícímu uživatelskému jménu. Na přihlašovací stránku se také směruje samotný systém, a to v případě, že se uživatel odhlásí.

Administrátor a redaktor

Do systému jsem implementoval dva právní typy, které jsem pracovně nazval „administrátor“ pro kompletní moc nad aplikací, a „redaktor“ pro omezenou moc. Administrátor může dělat v systému cokoliv, redaktor je omezen na manipulaci s obsahem (v případě blogu články), práci s kategoriemi a vysypávání koše. Články má povoleno publikovat. Pro grafické odlišení redaktor jednoduše nevidí v hlavním menu ty položky, se kterými nemá práva manipulovat. V tabulce níže jsou přesně vymezené pravomoci vysvětleny blíže.

Tab. 1: Přehled pravomocí uživatelských rolí systému

	Administrátor	Redaktor
Správa obsahu	**Ano**	**Ano**
Správa kategorií	**Ano**	**Ano**
Správa smazaného obsahu	**Ano**	**Ano**
Správa komentářů	**Ano**	**Ne**
Správa uživatelů	**Ano**	**Ne**
Nastavení systému	**Ano**	**Ne**

Systémový sledovač

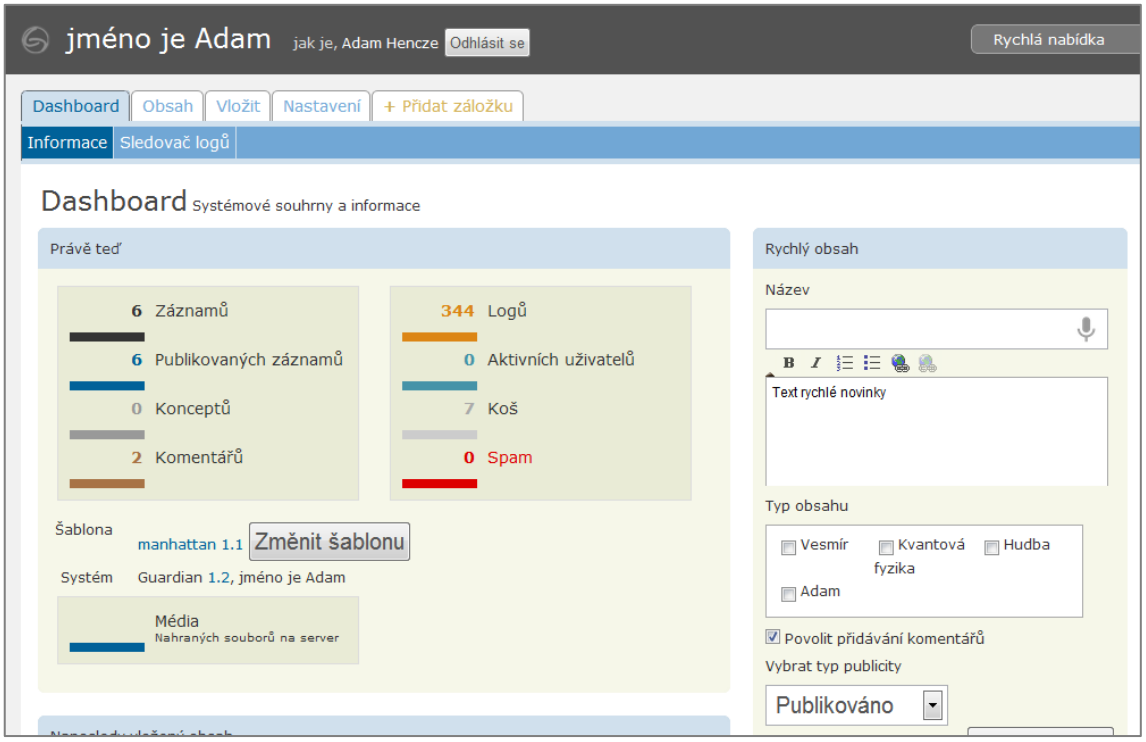
Napsal jsem funkci obecné kontroly všeho, co uživatel v rámci administrace provede. V databázi je pro to určena speciální tabulka, ve které jsou záznamy o aktivitě každého uživatele systému. Tento sledovač lze číst i přímo v grafickém prostředí administrace a slouží výhradně pro kontrolní účely, například když dojde ke smazání nějakého obsahu a správce neví, co se s ním stalo. Celý systém se tak archivuje.

Hodnoty ve sledovači nelze žádným způsobem upravovat ani mazat, jsou pouze pro čtení.

Systémové přehledy

Jsou hlavní stránkou systému, jsou na nich zobrazené statistiky a přehled obsahu. Nazval jsem jí Dashboard (česky „Palubní deska“) a je první věcí, kterou správce v systému uvidí. Samotné prostředí jsem designoval tak, aby vypadalo přehledně a zároveň líbivě. Nahoře je hlavní informační lišta, kde se vedle loga systému zobrazuje název blogu, který je zároveň odkazem do šablony. Vpravo od něj je jméno právě přihlášeného uživatele s tlačítkem pro odhlášení. V pravém horním rohu je ještě rolovací menu rychlé nabídky, které slouží jako seznam rychlých odkazů do systému.

Pod lištou se nachází hlavní menu administrace, které je dynamické a sestává ze systémové kategorie, do které je možné přidávat položky (lze i přímo v menu pomocí odkazu „přidat záložku“). Ve výchozím nastavení se jedná o přehledy, správu obsahu, vkládání obsahu a nastavení systému. Pod tímto elementem se nachází výběrové menu, které je závislé na hlavním menu a slouží pro výběr obsahu. Samotné systémové přehledy obsahují počet záznamů, konceptů, komentářů, logů (tedy záznamů ve sledovači), uživatelů, položek v koši, podezřelých spamů a médií. Zobrazují se i nejoblíbenější články a nové komentáře. V pravém sloupci je funkce rychlého obsahu.



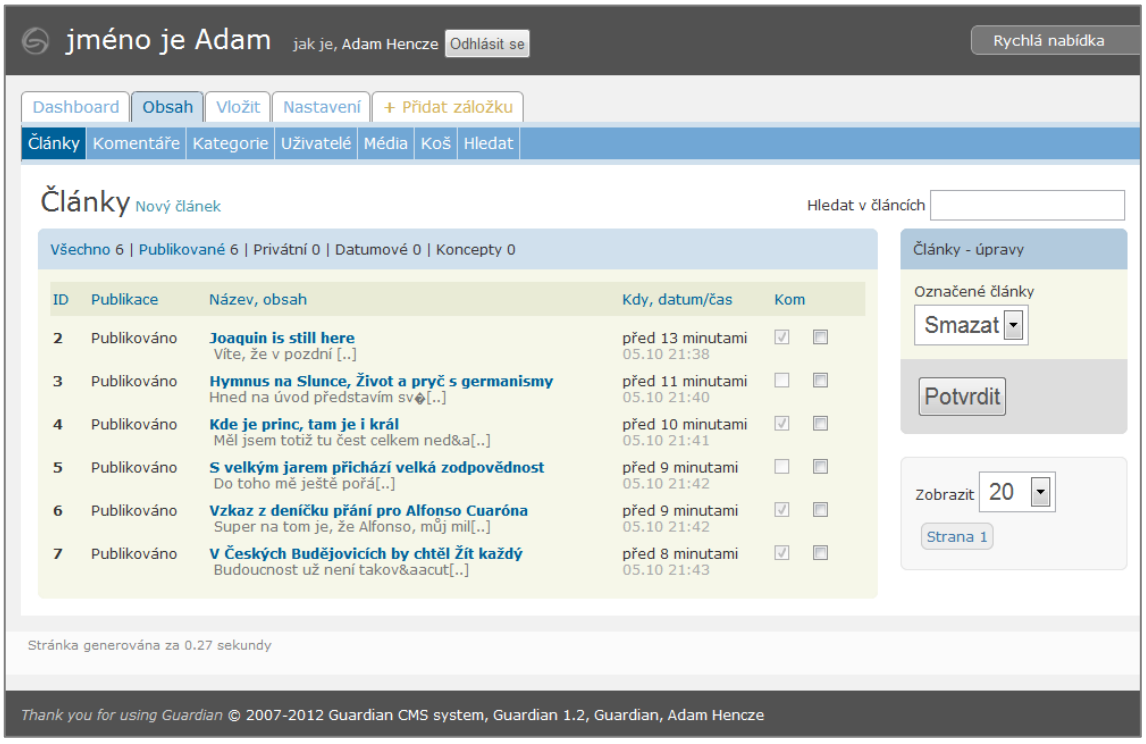
Obrázek 6: Hlavní stránka administrace

Rychlý obsah

Už podle názvu lze poznat, že slouží jako rychlé vložení nějaké krátké zprávy nebo krátkého článku na blog. Je vhodné tuto funkci použít (a je k tomu i navržena), když autor pospíchá a nechce se zabývat vyplňováním klíčových slov, dostávat se na vložení obsahu, používání prázdného konceptu apod. Ve spěchu pouze přejde do administrace, napíše rychlý text do textového pole, vyplní název, zaškrtne kategorii a obsah ihned vloží do databáze. Je to vhodné i pro vkládání konceptů, když autora napadne dobrý název nebo text k nějakému novému článku, může ho sem napsat a vložit jako nepublikovaný, tím ho bude mít připravený pro budoucí úpravy a vydání.

Zobrazení obsahu

Obsah je zobrazen jako jednoduchý seznam položek se základními informacemi, jako je název, úvodní obsah, datum vložení, povolení komentářů apod. Na výpisu lze s články manipulovat – lze je přímo upravovat, mazat a filtrovat. Toto zobrazení s těmito funkcemi jsou implementovány pomocí technologie AJAX, takže při například při filtrování se stránka nenačte znovu, pouze se aktualizuje seznam, což je pro práci efektivní.



Obrázek 7: Zobrazení článků v administraci

Práce s multimédii

Napsal jsem subsystém pro nahrávání souborů na disk serveru, který slouží pro obrázky. Tato část je potenciálním bezpečnostním rizikem, jelikož se soubor nahrává přímo na server, kde zůstane a kde ho lze spustit (pokud je spustitelný). Ve velké části případů zneužití systémů figurovala právě tato funkce, která měla neošetřený vstup. Vychytralý uživatel si tímto způsobem na server přes formulářové rozhraní nahrál například PHP skript, který umožňoval kontrolu nad systémem a jeho případné ovládnutí.

Systém pro nahrávání souborů je tak velmi striktní. Aby se soubor na server vůbec nahrál, ověřuji jeho typ (musí se jednat o obrázek typu „`image/jpg`“ nebo „`image/png`“) a jeho velikost (povolená velikost je do 250 kB). Pokud tyto kritéria splňuje, soubor se na server nahraje a do databáze se vloží jeho adresa, která se potom využívá při vložení média do obsahu článku.

Vyhledávání

Po několika nápadech jsem se nakonec rozhodl udělat vyhledávání fulltextové (tedy vyhledávání v obsahu porovnáváním jednotlivých slov se vstupním slovem). Po zadání vstupního slova se nejprve provede vyhledávání v názvech dokumentů, článků, v klíčových slovech a všech ostatních typů obsahu, teprve poté se provede vyhledávání fulltextem.

Výstup je potom v grafickém prostředí administrace vzhledově rozlišen, aby uživatel věděl, co přesně systém našel, jestli se jedná o článek, o klíčové slovo nebo přímo o slovo v obsahu některého z článků.

Koš

Je realizovaný jako zvláštní typ obsahu, který je uživatelem označen ke smazání. Stejně, jako se tato metoda osvědčila v dnešních operačních systémech, implementoval jsem ji i do svého systému. Veškerý smazaný obsah, od článků po komentáře, se ocitá v koši. Koš lze vysypat celý, nebo z něj nadobro smazat položky jednotlivě. Po této akci se obsah smaže i z databáze – to je důvod, proč jsem k vymazání položky napsal hodně kódu Javascriptu na hlášky typu „Jste si opravdu jisti smazáním této položky? Klikněte na Ano pro konečné smazání“, aby nedošlo k nechtěné chybě. Záznamy lze z koše samozřejmě obnovit zpět do jejich původního umístění.

Dodatek – šifrovací metody MD5 a SHA1

Jde o hashovací funkce, které jsou založené na matematických algoritmech. Jejich nejznámější vlastností je, že malá změna vstupních dat generuje velice odlišný výstupní hash.

Funkce MD5 generuje 128 bitů výstupního hashe, ale jsou s ní problémy. Její tvůrci odhalili malý problém [5] už v samotném návrhu funkce, nedávno se ale přišlo na mnohem větší bezpečnostní problémy, když tvůrci úspěšně nasimulovali kolizi. Prakticky šlo o dva různé vstupy, které generovaly stejný výstup, což je obrovské bezpečnostní riziko, proto se od používání této funkce dnes už opouští.

To je taky důvod, proč jsem se přes MD5 rozhodl implementovat funkci SHA1, která generuje 160 bitů [12] a je bezpečnější než MD5. Obecně platí čím víc bitů u hashovací funkce, tím bezpečnější je a tím déle trvá její prolomení (které trvá ideálně nekonečně dlouho, tento luxus ale dnešní hashovací funkce neumožňují).

3.3 Uživatelská dokumentace

Uživatelská dokumentace bývá psána přímo a jednoduše, může být buďto ve formě tzv. tutoriálu, nebo se může například jednat o prostý výčet funkcí a jejich popis. Podle dokumentace by také uživatel měl být schopen řešit problémy.

Napsal jsem tedy jednoduchou dokumentaci, která systém popisuje a pomocí které se v něm dokáže uživatel lépe zorientovat. Použil jsem grafický prvek, který sjednocuje všechny funkce a dává jim jednoduchý popis, říká, která role funkci smí používat apod. Dokumentaci jsem rozdělil do kapitol podobně jako tuto bakalářskou práci, s jednoduchým nástinem toho, co vlastně CMS systém je, jak funguje Guardian a potom samotný výčet jeho funkcí.

4 Vyhodnocení

Vývoj aplikace dospěl do části testování, kdy jsem testoval sám a dal systém i několika nezávislým testovacím uživatelům, kteří všechny funkce vyzkoušeli. Z vlastních zkušeností vím, že na většinu chyb přijde uživatel, který systém vidí nově, vývojář nikoliv. Ladící část byla dlouhá a nezáživná, ale u vývoje podobných aplikací jde o přirozenou součást, kde se chyby v určitém množství vyskytnou. Výsledky jsou ve finále uspokojivé, aplikace běží svižně (samozřejmě v závislosti na serveru, ale její požadavky na výkon jsou tak nízké, že na tom téměř nezáleží) a po odladění bez chyb.

4.1 Bezpečnost

Je jednou z mých hlavních priorit. Aplikace je bezpečná v rámci svých možností, používá jednosměrné šifrovací metody, co se týče citlivých dat uživatele a využívá hashovací funkce. Všechny akce v administraci se pokaždé kontrolují s uživatelskými údaji a je tak předcházeno nějakému pokusu o zneužití systému nebo údajů uživatele.

Konfigurační soubor systému, kde jsou citlivé údaje o přístupu do databáze, jména a hesla je chráněn pomocí serverového souboru .htaccess, což je vlastně doplňkový konfigurační soubor serveru, který se vkládá do adresáře, ve kterém mají jeho podmínky platit (ty pak platí i pro všechny podadresáře). Pomocí tohoto souboru se dá zakázat přístup určité IP[12] adresy, maskovat URL adresa apod. Tento soubor zakazuje přístup k systémovému konfiguračnímu souboru všem, kromě PHP stroje, který z něj čte.

Implementované vyhodnocování spamových komentářů a obsahu je rovněž plus pro bezpečnost systému. Jeho algoritmus není tak sofistikovaný, aby odchytil všechny spamové komentáře, může ale správce systému informovat o podezřelém obsahu.

Bezpečnost systému by šla ještě vylepšit implementací SSL[13] spojení tak, aby celá komunikace probíhala bezpečnostním protokolem HTTPS[14] přes ověřovací certifikát, který by byl podepsán autoritou (nebo sebou samým, to vyžaduje potvrzení výjimky při návštěvě systému). Tímto způsobem by byla aplikace chráněna před

[12] Internet Protocol, v češtině „Internetový Protokol“
[13] Secure Sockets Layer, v češtině „Vrstva Bezpečných Socketů“
[14] HyperText Transfer Protocol Secure, v češtině „Zabezpečený protokol přenosu hypertextu“

odposlouchávám informací, tuto funkci jsem ale nakonec neimplementoval, protože nebyla zadáním mé bakalářské práce.

4.2 Srovnání možností s ostatními systémy

V přímém srovnání s ostatními systémy, které jsem zkoumal a jejichž vlastnosti jsem v rešeršní části popisoval, můj systém neobstojí. Tyto systémy jsou spravovány stovkami vývojářů, teamy, které se zaobírají jejich různými částmi. Mým zadáním ale nebylo vytvořit nejlepší CMS systém na trhu, měl jsem si z těchto existujících systémů vzít to nejlepší a stvořit z toho svůj vlastní, jednoduchý systém, který cílí na běžné uživatele počítače.

V tomto ohledu má aplikace splňuje většinu základních funkcí ostatních systémů, včetně podpory různých šablon vzhledu. Nemá implementovaný správce doplňků, a jelikož jsem prozatím jediný systémový vývojář, nemělo by ani smysl ho do systému dopisovat (doplňky si píši sám), je to ale rozhodně primární cíl ke zlepšení aplikace. Nastavení systému jsem realizoval a aplikace lze konfigurovat podle libosti v rámci jejích možností. Složitost většiny profesionálních systémů je velká a proto také nabízí mnohem větší možnosti nastavení, než můj CMS.

Závěr

V bakalářské práci popisuji mnou navržený a zrealizovaný CMS systém, na který jsem použil podobné funkce a postupy z dnes již dostupných hotových řešení. Podařilo se mi tyto funkce implementovat takovým způsobem, že společně tvoří jednoduchý a přesto účinný systém, do kterého pro jeho intuitivnost pronikne i neznalec všech možných internetových technologií, což byl můj prvotní záměr. V závěru jsem možnosti mnou vytvořené aplikace porovnal s ostatními CMS systémy. Všechny body zadání se mi podařilo úspěšně splnit a k řešení jsem paradoxně používal technologie, které jsem se ve školních předmětech učil pouhý jeden semestr, podařilo se mi v nich ale zdokonalit na takovou úroveň, že jsou dnes mou hlavní pracovní náplní.

Nejdůležitějším výsledkem je, že systém běžel jako startovní řešení u dvou klientů s návštěvností průměrně padesát uživatelů denně. Během té doby se mi podařilo odladit určité množství chyb a získat důležitou zpětnou vazbu, podle které jsem systém vylepšil dle zkušeností zákazníka.

Seznam literatury

[1] BORONCZYK, Timothy. *PHP 6, MySQL, Apache – vytváříme webové aplikace*. Computer Press, 2009. 816 s. ISBN 978-80-251-2767-4.

[2] GILFILLAN, Ian. *Myslíme v MySQL 4*. Praha: GRADA Publishing, 2003. 750 s. ISBN 80-247-0661-X.

[3] GILMORE, W. J. *Velká kniha PHP a MySQL 5*. Brno: Zoner Press, 2007. 864 s. ISBN 80-86815-53-6.

[4] *Jak psát web*. [online]. 1998 [cit. 2012-05-08]. Dušan Jankovský. Dostupné z WWW: <www.jakpsatweb.cz>.

[5] *MD5 Hash Algorithm - Version 1.0*. [online]. 1997 [cit. 2012-05-10]. Philip A. DesAutels. Dostupné z WWW: <http://www.w3.org/TR/1998/REC-DSig-label/MD5-1_0>

[6] RAHMEL, Dan. *Joomla!* Computer Press, 2010. 384 s. ISBN 978-80-251-2714-8

[7] *SHA1 Secure Hash Algorithm - Version 1.0*. [online]. 1997 [cit. 2012-05-10]. Philip A. DesAutels. Dostupné z WWW: <http://www.w3.org/PICS/DSig/SHA1_1_0.html>

[8] *WordPress*. [online]. 1995 [cit. 2012-05-10]. WordPress. Dostupné z WWW: < http://wordpress.org/about/ >